SCHEDULING WITH FIXED DELIVERY DATES

NICHOLAS G. HALL

Department of Management Sciences, Fisher College of Business, The Ohio State University, Columbus, Ohio 43210-1144, hall.33@osu.edu

'MASEKA LESAOANA

Labour Market Information & Statistics, Department of Labour, Private Bag X117, Pretoria, South Africa 0001, maseka.Lesaoana@labour.gov.za

CHRIS N. POTTS

Faculty of Mathematical Studies, University of Southampton, Highfield, Southampton, United Kingdom, SO9 5N4, c.n.potts@maths.soton.ac.uk

(Received January 1997; revisions received September 1998, August 1999; accepted September 1999)

In most classical scheduling models, it is assumed that a job is dispatched to a customer immediately after its processing completes. In many practical situations, however, a set of delivery dates may be fixed before any jobs are processed. This is particularly relevant where delivery is an expensive or complicated operation, for example, as with heavy machinery. A similar situation arises where customers find deliveries disruptive and thus require them to be made within a limited time interval that repeats periodically. A third possibility is that a periodic business function, for example, the supplier's billing cycle, effectively defines a delivery date, and includes all jobs that have been completed since the previous billing cycle. These situations are not adequately represented by classical scheduling models. We consider a variety of deterministic scheduling problems in which a job is dispatched to a customer at the earliest fixed delivery date that is no earlier than the completion time of its processing. Problems where the number of delivery dates is constant, and others where it is specified as part of data input, are studied. For almost all problems considered, we either provide an efficient algorithm or establish that such an algorithm is unlikely to exist. By doing so, we permit comparisons between the solvability of these fixed delivery date problems and of the corresponding classical scheduling problems.

A standard assumption used in most of the classical scheduling literature is that a job is dispatched to a customer at the instant the job's processing is completed. However, several practical situations can be observed that are not adequately represented by this classical assumption about dispatch dates. In these situations, a job is dispatched at the earliest of a series of *fixed delivery dates*, which falls at or after the completion of its processing. The purpose of this paper is to investigate the solvability of scheduling problems in which this alternative definition of dispatch dates applies. These situations include the following:

- Supply chain management, where either economies of scale in delivery cost or the preferences of a downstream decision maker determine a list of possible delivery dates before scheduling issues are considered;
- Heavy engineering, where the delivery of a job may be an expensive or complicated operation that requires planning before decisions about the job processing sequence and timing have been made;
- Disruption of a customers' operations caused by a delivery, and the customer therefore limits the times during which deliveries can be made. Thus, deliveries are permitted only within a time interval that repeats periodically.
- A periodic business function, for example, the supplier's billing cycle, effectively defines a fixed delivery date. Here, economies of scale require the consolidation of bills for each period within a cyclic pattern, as discussed in the following paragraph. Another example of such consolidation is the mail collection times provided by the United States Postal Service.

Apparently, the earliest reference to fixed delivery dates in the literature is by Matsuo (1988), who considers a single machine environment. He points out that in many manufacturing environments, the number of shipping times is far less than the number of jobs. He provides an example in which 200 jobs are shipped over a 40-day planning horizon with 1 shipping time per day. Consolidation of shipments in this way results in substantial cost savings in transportation and handling. He considers the objective of finding an overtime utilization level and a job sequence that jointly provide a good trade-off between overtime cost and penalties for late delivery. A heuristic, based on a capacitated transshipment formulation, is described and tested computationally for this intractable problem. Lesaoana (1991) also considers problems with fixed delivery dates, and provides algorithms and computational complexity results for several different scheduling environments and objectives.

We consider a variety of the most practically and theoretically important nonpreemptive problems from the deterministic scheduling literature. We assume that a job is dispatched to a customer at the earliest fixed delivery date that

Subject classifications: Production/scheduling: scheduling with fixed delivery dates. Sequencing, deterministic: algorithms and complexity results.
Area of review: Manufacturing Operations.

is no earlier than the completion time of its processing. We study problems where the number of delivery dates is constant and also other problems where it is specified as part of the input for a particular instance. For almost all problems considered, we provide either an efficient algorithm or a proof that such an algorithm is unlikely to exist.

This paper is organized as follows. In §1 we introduce some basic definitions and notation and provide an overview of our results. Section 2 gives some general results that apply across several different scheduling environments or objectives. In §3 we study single machine problems. Section 4 considers identical parallel machine problems. Flow shops, job shops, and open shops are studied in §5, 6, and 7, respectively. Finally, §8 contains a conclusion and some suggestions for future research.

1. PRELIMINARIES

1.1. Notation and Classification

Let $N = \{1, \ldots, n\}$ denote the set of jobs to be processed nonpreemptively on m machines, $M_1, \ldots, M_m$. Let $p_{ij} \geq 1$ denote the processing time of job j on machine M_i, for $i = 1, \ldots, m$, $j = 1, \ldots, n$. In single and identical parallel machine problems, the first subscript is omitted. Where no ambiguity will arise, we may write $\sum_{j=1}^{n} p_{1j}$, $\sum_{j=1}^{n} p_{2j}$, and $\sum_{j=1}^{n} p_j$ as $\sum p_{1j}$, $\sum p_{2j}$, and $\sum p_j$, respectively. Let $D_1, \ldots, D_s$ denote the fixed delivery dates, where $0 < D_1 < \cdots < D_s$. In many applications, the time intervals between consecutive pairs of delivery dates are equal. However, we do not treat such situations separately because we have not identified any cases where this property makes a problem easier to solve. We consider problems where s is a fixed constant and also more general problems where s is arbitrary, i.e., part of the input. Other parameters of job j that occur in some problems include a due date d_j and a weight or value w_j. We assume throughout that all p_{ij}, d_j, w_j, and D_k are nonnegative integers.

We define the following variables in schedule σ.

$\widehat{C}_j(\sigma) =$ the time at which job j is dispatched to its customer;

$\widehat{L}_j(\sigma) = \widehat{C}_j(\sigma) - d_j$, the lateness of job j;

$$\widehat{U}_j(\sigma) = \begin{cases} 1 & \text{if job } j \text{ is late, i.e., } \widehat{C}_j(\sigma) > d_j, \\ 0 & \text{if job } j \text{ is dispatched to its customer by its due date, i.e., } \widehat{C}_j(\sigma) \leq d_j; \end{cases}$$

$\widehat{T}_j(\sigma) = \max\{\widehat{C}_j(\sigma) - d_j, 0\}$, the tardiness of job j.

When there is no ambiguity, we simplify $\widehat{C}_j(\sigma), \widehat{L}_j(\sigma)$, $\widehat{U}_j(\sigma)$, and $\widehat{T}_j(\sigma)$ to $\widehat{C}_j$, $\widehat{L}_j$, $\widehat{U}_j$, and $\widehat{T}_j$, respectively. We base performance measures on functions of $\widehat{C}_j(\sigma)$ because these dispatch times represent completion times as perceived by the customer. In a similar classical scheduling problem, the equivalent variables are C_j, L_j, U_j and T_j, respectively.

The standard classification scheme for scheduling problems (Graham et al. 1979) is $\psi_1|\psi_2|\psi_3$, where ψ_1 indicates the scheduling environment, ψ_2 describes the job characteristics or restrictive requirements, and ψ_3 defines the objective function to be minimized. We let $\psi_1 = \alpha m$, where $\alpha \in \{P, F, J, O\}$ denotes identical parallel machine, flow shop, job shop, and open shop environments, respectively, with m machines. If m is not shown, then the number of machines is arbitrary. If $m = 1$, then we consider a single machine environment and omit α. If $\psi_1 = Jm$, then we assume that each job has no more than one operation on any machine, for $m \geq 2$.

Under ψ_2, we may have:

$s = \bar{s}$: the number of delivery dates is a constant;

s: the number of delivery dates is arbitrary, i.e., specified in the input.

The objective functions we consider under ψ_3 require the minimization of the following cost functions:

$\widehat{C}_{\max} = \max_{1 \leq j \leq n}\{\widehat{C}_j\}$, the dispatch time of the last job, or the makespan;

$\sum(w_j)\widehat{C}_j =$ the total (weighted) dispatch time of the jobs;

$\widehat{L}_{\max} = \max_{1 \leq j \leq n}\{\widehat{L}_j\}$, the maximum lateness of the jobs;

$\sum(w_j)\widehat{U}_j =$ the (weighted) number of jobs not dispatched by their due dates;

$\sum(w_j)\widehat{T}_j =$ the total (weighted) tardiness of the jobs.

In general, we use γ to denote an objective function for a classical problem and $\hat{\gamma}$ to denote the corresponding objective function for a problem with fixed delivery dates.

It is useful to define the following categories of objective functions:

$\mathcal{A}$. $\sum \hat{f}_j$, where $\hat{f}_j \in \{\widehat{C}_j, w_j\widehat{C}_j\}$;

$\mathcal{B}$. $\hat{f}_{\max}$, where $\hat{f}_{\max} \in \{\widehat{C}_{\max}, \widehat{L}_{\max}\}$;

$\mathcal{C}$. $\sum \hat{f}_j$, where $\hat{f}_j \in \{\widehat{U}_j, w_j\widehat{U}_j\}$;

$\mathcal{D}$. $\sum \hat{f}_j$, where $\hat{f}_j \in \{\widehat{T}_j, w_j\widehat{T}_j\}$.

We use the convention that $f_j(D_k)$ denotes the cost of scheduling job j for delivery at time D_k, for any objective in $\mathcal{A}, \mathcal{B}, \mathcal{C}$, or $\mathcal{D}$, for $j \in N$. For any time $t > 0$, we let $|\lceil t \rceil|$ denote the earliest delivery date no earlier than t, and $|\lfloor t \rfloor|$ denote the latest delivery date no later than t. The time, $C_j(\sigma)$, at which the last operation of job j completes processing defines that job's dispatch time to be $\widehat{C}_j(\sigma) = D_k$, where k is such that $D_k = |\lceil C_j(\sigma) \rceil|$, or equivalently $D_{k-1} < C_j(\sigma) \leq D_k$, where $D_0 = 0$. If $C_j(\sigma) > D_s$, then we define $\widehat{C}_j(\sigma) = \infty$. For single machine scheduling problems and flow shops, if $\widehat{C}_j(\sigma) = D_k$, then job j is said to be *within block k*. For parallel machine scheduling problems, job shops, and open shops, an operation that is processed on some machine M_i is *within block k on* M_i if the time t at which it completes processing is such that $D_k = |\lceil t \rceil|$.

Table 1. Complexity of scheduling problems with fixed delivery dates.

Problem $\alpha \| \hat{\gamma}$	Complexity of $\alpha \| s = \bar{s} \| \hat{\gamma}$	Complexity of $\alpha \| s \| \hat{\gamma}$
$1 \| \widehat{C}_{\max}$	$O(n)$ Thm 5	$O(n)$ Thm 5
$1 \| \sum \widehat{C}_j$	$O(n)$ Thm 11	$O(\min\{n \log n, ns\})$ Thm 11
$1 \| \sum w_j \widehat{C}_j$	$BNPC$, $O(n(\sum p_j)^{\bar{s}-1})$ Thm 12,10	$UNPC$ Thm 13
$1 \| \widehat{L}_{\max}$	$O(n)$ Thm 14	$O(\min\{n \log n, ns\})$ Thm 14
$1 \| \sum \widehat{U}_j$	$O(n \log n)$ Cor 1	$O(n \log n)$ Cor 1
$1 \| \sum w_j \widehat{U}_j$	$BNPC$, $O(n \sum p_j)$ Cor 2, 1	$BNPC$, $O(n \sum p_j)$ Cor 2,1
$1 \| \sum \widehat{T}_j$	$BNPC$, $O(\min\{n(\sum p_j)^{\bar{s}-1}, n^4 \sum p_j\})$ Thm 15, 16	$BNPC$, $O(n^4 \sum p_j)$ Thm 15, 16
$1 \| \sum w_j \widehat{T}_j$	$BNPC$, $O(n(\sum p_j)^{\bar{s}-1})$ Thm 12, 4, 10	$UNPC$ Thm 13, 4
$Pm \| \hat{\gamma}$, $\hat{\gamma} \in \{\widehat{C}_{\max}, \widehat{L}_{\max}, \sum \widehat{C}_j\}$	$BNPC$, $O(n(\sum p_j)^{m-1})$ Thm 6, 3, 17	$BNPC$, $O(n(\sum p_j)^{m-1})$ Thm 6, 3, 17
$Pm \| \hat{\gamma}$, $\hat{\gamma} \in \{\sum \widehat{U}_j, \sum w_j \widehat{U}_j\}$	$BNPC$, $O(n(\sum p_j)^{m})$ Thm 6, 3, 17	$BNPC$, $O(n(\sum p_j)^{m})$ Thm 6, 3, 17
$Pm \| \sum \widehat{T}_j$	$BNPC$, $O(n(\sum p_j)^{m\bar{s}-1})$ Thm 6, 3, 18	$BNPC$, Open Thm 6, 3
$Pm \| \sum \hat{\gamma}$, $\hat{\gamma} \in \{w_j \widehat{C}_j, w_j \widehat{T}_j\}$	$BNPC$, $O(n(\sum p_j)^{m\bar{s}-1})$ Thm 6, 3, 18	$UNPC$ Thm 13, 4
$P \| \hat{\gamma}$	$UNPC$ Thm 6, 3	$UNPC$ Thm 6, 3
$F2 \| \widehat{C}_{\max}$	$O(n \log n)$ Thm 5	$O(n \log n)$ Thm 5
$F2 \| \hat{\gamma}$, $\hat{\gamma} \in \{\sum \widehat{C}_j, \sum w_j \widehat{C}_j, \widehat{L}_{\max}, \sum \widehat{T}_j, \sum w_j \widehat{T}_j\}$	$BNPC$, $O(n(\max\{\sum p_{1j}, \sum p_{2j}\})^{3\bar{s}})$ Thm 8, 4, 19, 20	$UNPC$ Thm 8, 4, 21
$F2 \| \hat{\gamma}$, $\hat{\gamma} \in \{\sum \widehat{U}_j, \sum w_j \widehat{U}_j\}$	$BNPC$, $O(n(\max\{\sum p_{1j}, \sum p_{2j}\})^{3\bar{s}+1})$ Thm 8, 4, 20	$UNPC$ Thm 8, 4
$F3 \| \hat{\gamma}$	$UNPC$ Thm 6, 3	$UNPC$ Thm 6, 3
$J2 \| \widehat{C}_{\max}$	$O(n \log n)$ Thm 5	$O(n \log n)$ Thm 5
$J2 \| \hat{\gamma}$, $\hat{\gamma} \in \{\sum \widehat{C}_j, \sum w_j \widehat{C}_j, \widehat{L}_{\max}, \sum \widehat{T}_j, \sum w_j \widehat{T}_j\}$	$BNPC$, $O(n^{2\bar{s}-1}(\max\{\sum p_{1j}, \sum p_{2j}\})^{18\bar{s}-12})$ Thm 8, 4, 19, 22	$UNPC$ Thm 8, 4, 21
$J2 \| \hat{\gamma}$, $\hat{\gamma} \in \{\sum \widehat{U}_j, \sum w_j \widehat{U}_j\}$	$BNPC$, $O(n^{2\bar{s}-1}(\max\{\sum p_{1j} \sum p_{2j}\})^{18\bar{s}-10})$ Thm 8, 4, 22	$UNPC$ Thm 8, 4
$J3 \| \hat{\gamma}$	$UNPC$ Thm 6, 3	$UNPC$ Thm 6, 3
$O2 \| \widehat{C}_{\max}$	$O(n)$ Thm 5	$O(n)$ Thm 5
$O2 \| \hat{\gamma}$, $\hat{\gamma} \in \{\sum \widehat{C}_j, \sum w_j \widehat{C}_j, \widehat{L}_{\max}, \sum \widehat{T}_j, \sum w_j \widehat{T}_j\}$	$BNPC$, $O(n^{2\bar{s}-1}(\max\{\sum p_{1j}, \sum p_{2j}\})^{15\bar{s}-11})$ Thm 8, 4, 23, 24	$UNPC$ Thm 8, 4, 25
$O2 \| \hat{\gamma}$, $\hat{\gamma} \in \{\sum \widehat{U}_j, \sum w_j \widehat{U}_j\}$	$BNPC$, $O(n^{2\bar{s}-1}(\max\{\sum p_{1j}, \sum p_{2j}\})^{15\bar{s}-9})$ Thm 8, 4, 24	$UNPC$ Thm 8, 4
$O3 \| \widehat{C}_{\max}$	$BNPC$, Open Thm 6	$BNPC$, Open Thm 6
$O \| \hat{\gamma}$	$UNPC$ Thm 6, 3	$UNPC$ Thm 6, 3

1.2. Overview of the Results

Table 1 provides a summary of our results. It is easy to see that the recognition versions of all the problems considered here belong to the class *NP*. We use *BNPC* (respectively, *UNPC*) to denote that the recognition version of a problem is binary (resp., unary) *NP*-complete. Related definitions can be found in Garey and Johnson (1979). Where a polynomial or pseudopolynomial time algorithm exists, we provide its time complexity. Included in the appropriate cell is a reference to where a proof of that result can be found.

In Table 1, some of the *BNPC* and *UNPC* entries for a problem $\alpha|s|\hat{\gamma}$ are deduced from the corresponding complexity results for problem $\alpha|s=\bar{s}|\hat{\gamma}$ because a problem with an arbitrary number of fixed delivery dates is more general than one with a constant number. Similarly, a property or algorithm for problem $\alpha|s|\hat{\gamma}$ also holds for problem $\alpha|s=\bar{s}|\hat{\gamma}$, which is a special case. In the following sections, such deductions are made without stating the result explicitly.

The relationship between classical and fixed delivery date problems is clarified by the following observation. Given an instance of a classical scheduling problem, we can construct an instance of the corresponding fixed delivery date problem in which each schedule has the same cost, by adding fixed delivery dates at every integer point in time. However, this observation cannot be used to imply complexity results because the size of the encoding of these delivery dates is not a polynomial function of the size of the input for the classical problem. Nonetheless, when the length of each interval between each consecutive pair of due dates is equal to the same constant, it would be possible to encode the delivery dates by specifying only D_1 and this constant, although doing so requires the use of an encoding scheme that is different from the standard one used throughout the paper.

Details of the proofs that are omitted from this paper or are only outlined can be found in a longer version (Hall et al. 1999).

2. GENERAL RESULTS

In this section we provide a number of results that apply to problems in several machine environments or with a variety of objectives.

2.1. Sequencing Within Blocks

The following two results identify problems for which the jobs assigned to a given machine and block may be sequenced using a simple rule.

THEOREM 1. *For problems $1|s|\hat{\gamma}$ and $Pm|s|\hat{\gamma}$, where $\hat{\gamma}$ is any objective defined in §1.1, there exists an optimal schedule in which the jobs within each block on a given machine are in an arbitrary sequence.*

THEOREM 2. *For problem $F2|s|\hat{\gamma}$, where $\hat{\gamma}$ is any objective defined in §1.1, there exists an optimal schedule in which the jobs within each block are sequenced on each machine according to the algorithm of Johnson (1954).*

2.2. Relationships Between Objectives

The following results describe relationships between the complexity of fixed delivery date scheduling problems with different objectives.

THEOREM 3. *If the recognition version of problem $\alpha|s=\bar{s}|\widehat{C}_{\max}$ is binary (respectively, unary) NP-complete, then the recognition version of problem $\alpha|s=\bar{s}|\hat{\gamma}$ is also binary (resp., unary) NP-complete, where $\hat{\gamma}$ is any other objective defined in §1.1.*

PROOF. There is an equivalence between the recognition version of problem $\alpha|s=\bar{s}|\widehat{C}_{\max}$ with a threshold cost of C and the recognition version of problem $\alpha|s=1|\hat{\gamma}$ with $D_1=||\lfloor C\rfloor||$ and (if applicable) $w_j=1$ and $d_j=D_1$ for $j=1,\ldots,n$, where the threshold cost C' is

$$C'=\begin{cases} nD_1 & \text{for } \hat{\gamma}=\sum(w_j)\widehat{C}_j, \\ 0 & \text{otherwise.} \end{cases}$$ □

Theorem 3 provides complexity results for several problems with fixed delivery dates that are binary or unary *NP*-complete from Theorem 6.

THEOREM 4. *If the recognition version of problem $\alpha|s=\bar{s}|\widehat{L}_{\max}$ (or $\alpha|s|\widehat{L}_{\max}$) is binary (respectively, unary) NP-complete, then the recognition version of problem $\alpha|s=\bar{s}|\hat{\gamma}$ (or $\alpha|s|\hat{\gamma}$) is also binary (resp., unary) NP-complete, where $\hat{\gamma}\in\{\sum(w_j)\widehat{U}_j,\sum(w_j)\widehat{T}_j\}$. Furthermore, if the recognition version of problem $\alpha|s=\bar{s}|\sum(w_j)\widehat{C}_j$ (or $\alpha|s|\sum(w_j)\widehat{C}_j$) is binary (resp., unary) NP-complete, then the recognition version of problem $\alpha|s=\bar{s}|\sum(w_j)\widehat{T}_j$ (or $\alpha|s|\sum(w_j)\widehat{T}_j$) is also binary (resp., unary) NP-complete.*

Theorem 4 provides complexity results for several problems with fixed delivery dates, including the recognition versions of: problem $1|s=\bar{s}|\sum w_j\widehat{T}_j$, which is binary *NP*-complete from Theorem 12; problems $1|s|\sum w_j\widehat{T}_j$ and $Pm|s|\sum w_j\widehat{T}_j$, which are unary *NP*-complete from Theorem 13; problems $\alpha 2|s=\bar{s}|\sum(w_j)\widehat{U}_j$ and $\alpha 2|s=\bar{s}|\sum(w_j)\widehat{T}_j$, where if $\alpha\in\{F,J,O\}$, the problems are binary *NP*-complete from Theorem 8; and problems $\alpha 2|s|\sum(w_j)\widehat{U}_j$ and $\alpha|s|\sum(w_j)\widehat{T}_j$, where if $\alpha\in\{F,J,O\}$, the problems are unary *NP*-complete from Theorem 8.

2.3. Makespan

The first result shows that any approach for solving a classical makespan problem can also be used to solve the corresponding makespan problem with fixed delivery dates.

THEOREM 5. *An algorithm that finds an optimal schedule for problem $\alpha\|C_{\max}$ also does so for problem $\alpha|s|\widehat{C}_{\max}$ with the same time complexity.*

From Theorem 5, the existence of a polynomial time algorithm for a classical makespan problem $\alpha\|C_{\max}$ implies that the same algorithm will solve the corresponding fixed delivery date problem in polynomial time. For example, an arbitrary sequence of the jobs solves problem $1|s|\widehat{C}_{\max}$ in

$O(n)$ time; an algorithm of Johnson (1954) solves problem $F2|s|\widehat{C}_{\max}$ in $O(n \log n)$ time; an algorithm of Jackson (1956) solves problem $J2|s|\widehat{C}_{\max}$ in $O(n \log n)$ time; and an algorithm of Gonzalez and Sahni (1976) solves problem $O2|s|\widehat{C}_{\max}$ in $O(n)$ time.

THEOREM 6. *If the recognition version of any classical scheduling problem $\alpha\|C_{\max}$ is binary (respectively, unary) NP-complete, then the recognition version of problem $\alpha|s = \bar{s}|\widehat{C}_{\max}$ is binary (resp., unary) NP-complete, even if $\bar{s} = 1$.*

PROOF. There is an equivalence between the recognition version of problem $\alpha\|C_{\max}$ with a threshold cost of C, and the recognition version of problem $\alpha|s = 1|\widehat{C}_{\max}$ with $D_1 = C$ and a threshold cost of C. □

Theorem 6 provides complexity results for several makespan problems with fixed delivery dates, including the recognition versions of: problem $P2|s = \bar{s}|\widehat{C}_{\max}$, which is binary *NP*-complete (Karp 1972); problem $P|s = \bar{s}|\widehat{C}_{\max}$, which is unary *NP*-complete (Garey and Johnson 1978); problems $F3|s = \bar{s}|\widehat{C}_{\max}$ and $J3|s = \bar{s}|\widehat{C}_{\max}$, which are unary *NP*-complete (Garey et al. 1976); problem $O3|s = \bar{s}|\widehat{C}_{\max}$, which is binary *NP*-complete (Gonzalez and Sahni 1976); and problem $O|s = \bar{s}|\widehat{C}_{\max}$, which is unary *NP*-complete (Williamson et al. 1997). Using Theorem 3, corresponding results are deduced for the other objective functions defined in §1.1.

2.4. Maximum Lateness

We study the problem of minimizing the maximum lateness with an arbitrary number of fixed delivery dates, or problem $\alpha|s|\widehat{L}_{\max}$. The following result shows how an algorithm for the equivalent classical problem can be used here.

THEOREM 7. *An algorithm that finds an optimal schedule in polynomial or pseudopolynomial time for problem $\alpha\|L_{\max}$ also does so for problem $\alpha|s|\widehat{L}_{\max}$.*

PROOF. An algorithm for the recognition version of problem $\alpha\|L_{\max}$ with a threshold cost of zero can be used to determine whether there is a schedule for problem $\alpha|s|\widehat{L}_{\max}$ with $\widehat{L}_{\max} \leq L$ by setting due dates $|\lfloor d_j + L \rfloor|$ for $j = 1, \ldots, n$ in the classical problem. □

Although Theorem 7 establishes that problem $1|s|\widehat{L}_{\max}$ is solvable in polynomial time, we provide a more efficient algorithm in Theorem 14. The following result also provides a connection between classical problems and those with fixed delivery dates.

THEOREM 8. *If the recognition version of a classical scheduling problem $\alpha\|L_{\max}$ is binary (respectively, unary) NP-complete, then the recognition version of the corresponding problem $\alpha|s|\widehat{L}_{\max}$ is binary (resp., unary) NP-complete. Furthermore, if the recognition version of a special case of a classical problem $\alpha\|L_{\max}$ with a constant number of due dates is binary (resp., unary) NP-complete, then the recognition version of the corresponding problem $\alpha|s = \bar{s}|\widehat{L}_{\max}$ is binary (resp., unary) NP-complete.*

PROOF. The recognition version of problem $\alpha\|L_{\max}$ is equivalent to the recognition version of problem $\alpha|s|\widehat{L}_{\max}$ when delivery dates are equal to the distinct values of the due dates. □

Theorem 8 provides complexity results for several maximum lateness problems with fixed delivery dates, including the recognition versions of: problem $\alpha 2|s = \bar{s}|\widehat{L}_{\max}$, which is binary *NP*-complete for $\alpha \in \{F, J\}$ (Lenstra 1977) and $\alpha = O$ (by a straightforward adaptation of the reduction of Lawler et al. 1981, 1982); and problem $\alpha 2|s|\widehat{L}_{\max}$, which is unary *NP*-complete for $\alpha \in \{F, J\}$ (Garey et al. 1976), and for $\alpha = O$ (Lawler et al. 1981, 1982). Using Theorems 4 and 8, corresponding computational complexity results for the $\sum(w_j)\widehat{U}_j$ and $\sum(w_j)\widehat{T}_j$ objectives can be deduced.

2.5. Weighted Number of Late Jobs

Consider problem $\alpha|s|\sum w_j\widehat{U}_j$. The following result shows that there is an equivalent instance of the classical problem $\alpha\|\sum U_j$.

THEOREM 9. *Problems $\alpha|s|\sum w_j\widehat{U}_j$ and $\alpha\|\sum w_j U_j$ are equivalent, where an instance of the latter problem is constructed from the corresponding instance of the former by setting due dates $d'_j = |\lfloor d_j \rfloor|$ for $j = 1, \ldots, n$.*

COROLLARY 1. *An algorithm that finds an optimal schedule for problem $\alpha\|\sum(w_j)U_j$ also does so for problem $\alpha|s|\sum(w_j)\widehat{U}_j$ with the same time complexity.*

Corollary 1 implies that the algorithm of Moore (1968) solves problem $1|s|\sum\widehat{U}_j$ in $O(n \log n)$ time. Similarly, the algorithm of Lawler and Moore (1969) solves problem $1|s|\sum w_j\widehat{U}_j$ in $O(n\sum p_j)$ time.

COROLLARY 2. *If the recognition version of a classical scheduling problem $\alpha\|\sum(w_j)U_j$ is binary (respectively, unary) NP-complete, then the recognition version of the corresponding problem $\alpha|s|\sum(w_j)\widehat{U}_j$ is binary (resp., unary) NP-complete. Furthermore, if the recognition version of a special case of a classical problem $\alpha\|\sum(w_j)U_j$ with a constant number of due dates is binary (resp., unary) NP-complete, then the recognition version of the corresponding problem $\alpha|s = \bar{s}|\sum(w_j)\widehat{U}_j$ is binary (resp., unary) NP-complete.*

The work of Karp (1972) and Corollary 2 imply that the recognition version of problem $1|s = \bar{s}|\sum w_j\widehat{U}_j$ is binary *NP*-complete. We have now established the computational complexity of problems $1|s = \bar{s}|\sum(w_j)\widehat{U}_j$ and $1|s|\sum(w_j)\widehat{U}_j$.

3. SINGLE MACHINE

In this section we consider a number of the most widely studied single machine scheduling problems. Each subsection considers a different objective.

3.1. Total and Maximum Cost

We now describe a pseudopolynomial time dynamic programming algorithm, BLOCK, for problem $1|s = \bar{s}|\hat{\gamma}$, where $\hat{\gamma}$ is any objective in category $\mathcal{A}$, $\mathcal{B}$, or $\mathcal{D}$. It is useful to define the operator $\oplus$, where $\oplus$ represents the "max" operator for objectives in $\mathcal{B}$ and the "+" operator elsewhere. Recall from Theorem 1 that any sequence of jobs within a block can be used. Thus, Algorithm BLOCK considers jobs in an arbitrary order and evaluates the possible assignment of any job to each block.

ALGORITHM BLOCK

State variables:

In addition to a variable j that indicates that jobs $1, \ldots, j$ have been scheduled, we also use state variables $(t_1, \ldots, t_{\bar{s}})$, where t_k is the total processing time of jobs assigned to block k, for $k = 1, \ldots, \bar{s}$. When jobs $1, \ldots, j$ have been scheduled, we restrict the choice of state variables to satisfy $(t_1, \ldots, t_{\bar{s}}) \in \mathcal{T}_j$, where $\mathcal{T}_j = \{(t_1, \ldots, t_{\bar{s}}) | \sum_{h=1}^{k} t_h \leq D_k \text{ for } k = 1, \ldots, \bar{s}, \sum_{k=1}^{\bar{s}} t_k = \sum_{l=1}^{j} p_l\}$.

Optimal value function:

Let $g_j(t_1, \ldots, t_{\bar{s}})$ denote the minimum cost of scheduling jobs $1, \ldots, j$ such that the state $(t_1, \ldots, t_{\bar{s}})$ is achieved.

Recurrence relation:

$g_j(t_1, \ldots, t_{\bar{s}}) = \min_{k \in \{1, \ldots, \bar{s}\}} \{g_{j-1}(t_1, \ldots, t_{k-1}, t_k - p_j, t_{k+1}, \ldots, t_{\bar{s}}) \oplus f_j(D_k)\}$.

Boundary condition:

$g_0(0, \ldots, 0) = 0$, and all other values are set to ∞, for $\hat{\gamma} \neq \widehat{L}_{\max}$.
$g_0(0, \ldots, 0) = -\infty$, and all other values are set to ∞, for $\hat{\gamma} = \widehat{L}_{\max}$.

Optimal solution:

$$\min_{(t_1, \ldots, t_{\bar{s}}) \in \mathcal{T}_n} \{g_n(t_1, \ldots, t_{\bar{s}})\}.$$

THEOREM 10. *Algorithm BLOCK finds an optimal schedule for problem* $1|s = \bar{s}|\sum \hat{\gamma}$ *with time complexity* $O(n(\sum p_j)^{\bar{s}-1})$ *for objectives in* $\mathcal{A}$, $\mathcal{B}$, *and* $\mathcal{D}$.

PROOF. The time complexity is determined by the number of values for the state variables, which is $O(n(\sum p_j)^{\bar{s}-1})$ because $\sum_{k=1}^{\bar{s}} t_k = \sum_{h=1}^{j} p_h$. □

We note that Algorithm BLOCK can be also adapted for objectives in $\mathcal{C}$. However, Corollary 1 provides more efficient procedures.

3.2. Sum of Dispatch Times

The classical problem of minimizing the sum of job completion times on a single machine, or $1\|\sum C_j$, is optimized by the shortest processing time (SPT) rule that requires $O(n \log n)$ time. A similar approach can be used for the problem with fixed delivery dates. However, since jobs within any block have a common dispatch time, they can be processed in any sequence, which may permit a reduction in the time complexity.

THEOREM 11. *An optimal schedule for problem* $1|s|\sum \widehat{C}_j$ *can be found in* $O(\min\{n \log n, ns\})$ *time, and an optimal schedule for problem* $1|s = \bar{s}|\sum \widehat{C}_j$ *can be found in* $O(n)$ *time.*

PROOF. For problem $1|s|\sum \widehat{C}_j$, the completion time of the job in any given position is no larger for an SPT sequence than for any other sequence. Thus, an SPT schedule gives an optimal solution in $O(n \log n)$ time. Alternatively, linear time median-finding techniques (Blum et al. 1973, Schonhäge et al. 1976) allow an optimal partition of jobs into blocks in $O(ns)$ time. □

3.3. Sum of Weighted Dispatch Times

It is well known (Smith 1956) that the shortest weighted processing time (SWPT) rule gives an optimal solution to problem $1\|\sum w_j C_j$. We now provide a numerical example to show that the SWPT rule fails to optimize the corresponding fixed delivery date problem, $1|s = \bar{s}|\sum w_j \widehat{C}_j$.

EXAMPLE 1. *Let* $n = 3, \bar{s} = 2 : p_1 = 2, p_2 = p_3 = 4; w_1 = 3, w_2 = 5, w_3 = 4$*; and* $D_1 = 5$, $D_2 = 10$.

The unique SWPT sequence $S_1 = (1, 2, 3)$ has value 105, but the optimal solution value of 95 is given by non-SWPT sequences $S_2 = (2, 1, 3)$ and $S_3 = (2, 3, 1)$.

The following *NP*-completeness result uses a reduction from Partition (Garey and Johnson 1979).

THEOREM 12. *The recognition version of problem* $1|s = \bar{s}|\sum w_j \widehat{C}_j$ *is binary* NP-*complete.*

Algorithm BLOCK, described in §3.1, solves problem $1|s = \bar{s}|\sum w_j \widehat{C}_j$ in pseudopolynomial time where we set $f_j(D_k) = w_j D_k$ for $j = 1, \ldots, n, k = 1, \ldots, \bar{s}$. However, the following *NP*-completeness result, which uses a reduction from 3-Partition (Garey and Johnson 1979), shows that the existence of a pseudopolynomial time algorithm for the more general problem with an arbitrary number of delivery dates is unlikely.

THEOREM 13. *The recognition version of problem* $1|s|\sum w_j \widehat{C}_j$ *is unary* NP-*complete.*

3.4. Maximum Lateness

We consider problem $1|s|\widehat{L}_{\max}$. Recall that for the corresponding classical problem, $1\|L_{\max}$, the earliest due date (EDD) rule, in which jobs are sequenced in nondecreasing due date order, gives an optimal schedule (Jackson 1955). We show how to use an EDD ordering to solve problem $1|s|\widehat{L}_{\max}$.

THEOREM 14. *An optimal schedule for problem* $1|s|\widehat{L}_{\max}$ *can be found in* $O(\min\{n\log n, ns\})$ *time, and an optimal schedule for problem* $1|s=\bar{s}|\widehat{L}_{\max}$ *can be found in* $O(n)$ *time.*

PROOF. An optimal schedule is obtained in $O(n\log n)$ time by sequencing the jobs in EDD order. Alternatively, the jobs can be optimally partitioned into blocks in $O(ns)$ time using median-finding techniques, as in Theorem 11. □

3.5. Total (Weighted) Tardiness

The following *NP*-completeness result uses a reduction from Equal Cardinality Partition (Garey and Johnson 1979).

THEOREM 15. *The recognition version of problem* $1|s=\bar{s}|\sum\widehat{T}_j$ *is binary* NP*-complete.*

However, we are able to provide two pseudopolynomial time algorithms for total tardiness problems.

THEOREM 16. *An optimal schedule for problem* $1|s|\sum\widehat{T}_j$ *can be found in* $O(n^4\sum p_j)$ *time, and an optimal schedule for problem* $1|s=\bar{s}|\sum\widehat{T}_j$ *can be found in* $O(\min\{n(\sum p_j)^{\bar{s}-1}, n^4\sum p_j\})$ *time.*

PROOF. For both problems, the algorithm of Lawler (1977) can be adapted. Alternatively, for problem $1|s=\bar{s}|\sum\widehat{T}_j$, Theorem 10 shows that Algorithm BLOCK can be applied, where we set $f_j(D_k)=\max\{D_k-d_j,0\}$ for $j=1,\ldots,n, k=1,\ldots,\bar{s}$. □

4. PARALLEL MACHINES

In this section we develop two pseudopolynomial time dynamic programming algorithms for solving a variety of identical parallel machine problems.

The first algorithm considers the jobs in an appropriate sequence and assigns each in turn to a machine, or in the case of (weighted) numbers of late jobs problems may alternatively discard jobs. Let $\mathscr{A}'$ denote the restriction of $\mathscr{A}$ to $\sum\hat{f}_j=\sum\widehat{C}_j$. The algorithm solves problem $Pm|s|\hat{\gamma}$, where $\hat{\gamma}$ is any objective in $\mathscr{A}'$, $\mathscr{B}$, or $\mathscr{C}$. For the objective in $\mathscr{A}'$, the jobs are indexed in SPT order. For the objectives in $\mathscr{B}$, the jobs are indexed in arbitrary order for $\widehat{C}_{\max}$ and in EDD order for $\widehat{L}_{\max}$. For the objectives in $\mathscr{C}$, the jobs are indexed in EDD order.

ALGORITHM PARALLEL

State variables:

In addition to a variable j that indicates that jobs $1,\ldots,j$ have been scheduled, we also use state variables $(t_1,\ldots,t_m)$, where t_i is the total processing time of jobs scheduled on machine M_i, for $i=1,\ldots,m$. When jobs $1,\ldots,j$ have been scheduled, we restrict the choice of state variables to satisfy $(t_1,\ldots,t_m)\in\mathscr{T}_j$, where $\mathscr{T}_j=\{(t_1,\ldots,t_m)|\sum_{i=1}^m t_i=\sum_{l=1}^j p_l\}$ for objectives in $\mathscr{A}'$ and $\mathscr{B}$, and where $\mathscr{T}_j=\{(t_1,\ldots,t_m)|\sum_{i=1}^m t_i\le\sum_{l=1}^j p_l,\ t_i\le|\lfloor d_j\rfloor|$ for $i=1,\ldots,m\}$ for objectives in $\mathscr{C}$.

Optimal value function:

Let $g_j(t_1,\ldots,t_m)$ denote the minimum cost of scheduling jobs $1,\ldots,j$ such that the state $(t_1,\ldots,t_m)$ is achieved.

Recurrence relation:

$\mathscr{A}',\mathscr{B}$. $g_j(t_1,\ldots,t_m)=\min_{i\in\{1,\ldots,m\}}\{g_{j-1}(t_1,\ldots,t_{i-1},t_i-p_j,t_{i+1},\ldots,t_m)\oplus f_j(|\lceil t_i\rceil|)\}$;
$\mathscr{C}$. $g_j(t_1,\ldots,t_m)=\min\{\min_{i\in\{1,\ldots,m\}}\{g_{j-1}(t_1,\ldots,t_{i-1},t_i-p_j,t_{i+1},\ldots,t_m)\}, g_{j-1}(t_1,\ldots,t_m)+w_j\}$.

Boundary condition:

$g_0(0,\ldots,0)=0$, and all other values are set to ∞, for $\hat{\gamma}\ne\widehat{L}_{\max}$.
$g_0(0,\ldots,0)=-\infty$, and all other values are set to ∞, for $\hat{\gamma}=\widehat{L}_{\max}$.

Optimal solution

$$\min_{(t_1,\ldots,t_m)\in\mathscr{T}_n}\{g_n(t_1,\ldots,t_m)\}.$$

THEOREM 17. *Algorithm PARALLEL finds an optimal schedule for problem* $Pm|s|\hat{\gamma}$, *where* $\hat{\gamma}$ *is any objective in* $\mathscr{A}'$ *or* $\mathscr{B}$ *with time complexity* $O(n(\sum p_j)^{m-1})$, *and where* $\hat{\gamma}$ *is any objective in* $\mathscr{C}$ *with time complexity* $O(n(\sum p_j)^m)$.

We propose a dynamic programming algorithm, PBLOCK, for problem $Pm|s=\bar{s}|\hat{\gamma}$, where $\hat{\gamma}$ is any objective in $\mathscr{A}$, $\mathscr{B}$, or $\mathscr{D}$. Apart from an extended set of state variables $(t_{11},\ldots,t_{1\bar{s}},\ldots,t_{m1},\ldots,t_{m\bar{s}})$, where t_{ik} is the total processing time of jobs assigned to block k on machine M_i for $i=1,\ldots,m, k=1,\ldots,\bar{s}$, PBLOCK has exactly the same structure as Algorithm BLOCK in §3.1, and therefore a formal description is omitted.

THEOREM 18. *Algorithm PBLOCK finds an optimal schedule for problem* $Pm|s=\bar{s}|\hat{\gamma}$ *with time complexity* $O(n(\sum p_j)^{m\bar{s}-1})$ *for objectives in* $\mathscr{A}$, $\mathscr{B}$, *and* $\mathscr{D}$.

It is possible to adapt Algorithm PBLOCK for objectives in $\mathscr{C}$. However, Algorithm PARALLEL is more efficient.

A comment about problem $Pm|s|\sum\hat{T}_j$ appears in §8.

5. FLOW SHOPS

Recall that the recognition version of the classical problem $F2\|\sum C_j$ is unary *NP*-complete (Garey et al. 1976). However, we now demonstrate that the corresponding fixed delivery date problem is only binary *NP*-complete. The binary *NP*-completeness result is established using a reduction from Equal Cardinality Partition (Garey and Johnson 1979).

THEOREM 19. *The recognition version of problem* $F2|s=\bar{s}|\sum\widehat{C}_j$ *is binary NP-complete.*

We now describe a dynamic programming algorithm, FBLOCK, which finds an optimal schedule for problem $F2|s=\bar{s}|\hat{\gamma}$, where $\hat{\gamma}$ is any objective in $\mathscr{A}$, $\mathscr{B}$, $\mathscr{C}$, or $\mathscr{D}$.

In view of Theorem 2, we assume that the jobs are indexed according to the algorithm of Johnson (1954). Our

dynamic programming algorithm, FBLOCK, assigns jobs to blocks and sequences the jobs according to Johnson's algorithm. A complete schedule is obtained by concatenating the individual schedules for the blocks.

ALGORITHM FBLOCK

State variables:

In addition to a variable j that indicates that jobs $1, \dots, j$ have been scheduled, we also use state variables $(t_{11}, t_{21}, I_1, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}})$, where t_{ik} denotes the minimum time required to complete all jobs within block k on machine M_i, for $i = 1, 2$, and I_k denotes the idle time on machine M_2 in block k, for $k = 1, \dots, \bar{s}$. We associate with each state the value T_{ik}, which denotes the time that the last job of block k completes processing on machine M_i, for $i = 1, 2$, $k = 1, \dots, \bar{s}$. The values T_{ik} are related to the state variables through the equations $T_{11} = t_{11}$, $T_{21} = t_{21}$, $T_{1k} = T_{1,k-1} + t_{1k}$, and $T_{2k} = \max\{T_{1,k-1} + t_{2k}, T_{2,k-1} + t_{2k} - I_k\}$, for $k = 2, \dots, \bar{s}$. When jobs $1, \dots, j$ have been scheduled, we restrict the choice of state variables to satisfy $(t_{11}, t_{21}, I_1, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}}) \in \mathcal{T}_j$, where $\mathcal{T}_j = \{(t_{11}, t_{21}, I_1, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}}) | T_{2k} \le D_k, \sum_{k=1}^{\bar{s}} t_{1k} = \sum_{h=1}^{j} p_{1h}, I_k \le t_{1k}, t_{2k} \le \sum_{h=1}^{j} p_{2h} + I_k$, for $k = 1, \dots, \bar{s}\}$ for objectives in $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{D}$ and where $\mathcal{T}_j = \{(t_{11}, t_{21}, I_1, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}}) | T_{2k} \le D_k, \sum_{k=1}^{\bar{s}} t_{1k} \le \sum_{h=1}^{j} p_{1h}, I_k \le t_{1k}, t_{2k} \le \sum_{h=1}^{j} p_{2h} + I_k$, for $k = 1, \dots, \bar{s}\}$ for objectives in $\mathcal{C}$.

Optimal value function:

Let $g_j(t_{11}, t_{21}, I_1, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}})$ denote the minimum cost of scheduling jobs $1, \dots, j$ such that the state $(t_{11}, t_{21}, I_1, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}})$ is achieved.

Recurrence relation:

$$\mathcal{A}, \mathcal{B}, \mathcal{D}.\ g_j(t) = \min\Big\{\min_{k \in K_1} \big\{g_{j-1}(\dots, t_{1k} - p_{1j}, t_{2k} - p_{2j}, I_k, \dots) \oplus f_j(D_k)\big\},$$
$$\min_{k \in K_2,\, \tau \in \{t_{1k} - p_{1j}, \dots, t_{1k}\}} \big\{g_{j-1}(\dots, t_{1k} - p_{1j}, \tau, I_k - (t_{1k} - \tau), \dots) \oplus f_j(D_k)\big\}\Big\};$$

$$\mathcal{C}.\ g_j(t) = \min\Big\{\min_{k \in K_3} \big\{g_{j-1}(\dots, t_{1k} - p_{1j}, t_{2k} - p_{2j}, I_k, \dots) \oplus f_j(D_k)\big\},$$
$$\min_{k \in K_4,\, \tau \in \{t_{1k} - p_{1j}, \dots, t_{1k}\}} \big\{g_{j-1}(\dots, t_{1k} - p_{1j}, \tau, I_k - (t_{1k} - \tau), \dots) \oplus f_j(D_k)\big\}, g_{j-1}(t) + w_j\Big\},$$

where

$$t = (t_{11}, t_{21}, I_1, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}}),$$
$$(\dots, t_{1k} - p_{1j}, t_{2k} - p_{2j}, I_k, \dots)$$

is an abbreviation for $(t_{11}, t_{21}, I_1, \dots, t_{1,k-1}, t_{2,k-1}, I_{k-1}, t_{1k} - p_{1j}, t_{2k} - p_{2j}, I_k, t_{1,k+1}, t_{2,k+1}, I_{k+1}, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}})$ with $(\dots, t_{1k} - p_{1j}, \tau, I_k - (t_{1k} - \tau), \dots)$ defined analogously, $K_1 = \{k | k \in \{1, \dots, \bar{s}\}, t_{1k} + p_{2j} < t_{2k}\}$, $K_2 = \{k | k \in \{1, \dots, \bar{s}\}, t_{1k} + p_{2j} = t_{2k}\}$, $K_3 = \{k | k \in K_1, D_k \le \lfloor\lfloor d_j \rfloor\rfloor\}$, and $K_4 = \{k | k \in K_2, D_k \le \lfloor\lfloor d_j \rfloor\rfloor\}$.

Boundary condition:

$g_0(0, \dots, 0) = 0$,

and all other values are set to ∞, for $\hat{\gamma} \ne \widehat{L}_{\max}$.

$g_0(0, \dots, 0) = -\infty$,

and all other values are set to ∞, for $\hat{\gamma} = \widehat{L}_{\max}$.

Optimal solution:

$$\min_{(t_{11}, t_{21}, I_1, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}}) \in \mathcal{T}_n} \big\{g_n(t_{11}, t_{21}, I_1, \dots, t_{1\bar{s}}, t_{2\bar{s}}, I_{\bar{s}})\big\}.$$

In the recurrence relation, K_1 and K_3 correspond to a situation where the time t_{1k} at which job j completes processing on machine M_1 is earlier than the time $t_{2k} - p_{2j}$ at which it starts processing on machine M_2. In this case, job j does not generate any additional idle time. On the other hand, K_2 and K_4 correspond to a situation where job j is processed in no-wait mode. In this case, $t_{1k} = t_{2k} - p_{2j}$, and the previous schedule for block k completes processing on machine M_1 at time $t_{1k} - p_{1j}$ and on machine M_2 at some time τ, where $t_{1k} - p_{1j} \le \tau \le t_{2k} - p_{2j}$. During the interval $[\tau, t_{1k}]$, machine M_2 is idle.

THEOREM 20. *Algorithm FBLOCK finds an optimal schedule for problem $F2|s = \bar{s}|\hat{\gamma}$, where $\hat{\gamma}$ is any objective in $\mathcal{A}, \mathcal{B}$, or $\mathcal{D}$ with time complexity $O(n(\max\{\sum p_{1j}, \sum p_{2j}\})^{3\bar{s}})$, and where $\hat{\gamma}$ is any objective in $\mathcal{C}$ with time complexity $O(n(\max\{\sum p_{1j}, \sum p_{2j}\})^{3\bar{s}+1})$.*

The following *NP*-completeness result uses a reduction from 3-Partition (Garey and Johnson 1979).

THEOREM 21. *The recognition version of problem $F2|s|\sum \widehat{C}_j$ is unary* NP*-complete.*

6. JOB SHOPS

In this section we describe the main features of a dynamic programming algorithm, JBLOCK, for several job shop problems with a constant number of fixed delivery dates. There are four types of jobs in problem $J2|s|\hat{\gamma}$, where $\hat{\gamma}$ is any objective in $\mathcal{A}, \mathcal{B}, \mathcal{C}$, or $\mathcal{D}$. We let J_i denote the set of jobs that require processing on machine M_i only, for $i = 1, 2$, and we let J_{12} (respectively, J_{21}) denote the set of jobs that require processing on machine M_1 followed by machine M_2 (resp., machine M_2 followed by machine M_1).

Block k of a given schedule, for $k = 1, \dots, \bar{s}$, is partitioned into *groups* of jobs as follows. Let J_{12k}^1 (respectively, J_{21k}^1) denote the jobs of J_{12} (resp., J_{21}) that have their first operation within block k and their second operation within a later block. Let J_{12k}^2 (respectively, J_{21k}^2) denote the jobs of J_{12} (resp., J_{21}) that have their second operation within block k and their first operation within an earlier block. Let J_{12k}^0 (respectively, J_{21k}^0) denote the jobs of J_{12} (resp., J_{21}) that have both operations within block k. Finally, let J_{1k} (respectively, J_{2k}) denote the jobs of J_1 (resp., J_2) that have their single operation within block k.

Algorithm JBLOCK relies on various properties of an optimal schedule. First, the sequence of groups of block k on machine M_1 is $(J^0_{12k}, J^1_{12k}, J_{1k}, J^2_{21k}, J^0_{21k})$, and the sequence of groups of block k on machine M_2 is $(J^0_{21k}, J^1_{21k}, J_{2k}, J^2_{12k}, J^0_{12k})$, for $k = 1, \ldots, \bar{s}$. Accordingly, we refer to $J^0_{12k}, J^1_{12k}, J_{1k}, J^2_{21k}, J^0_{21k}$ as groups 1, 2, 3, 4, 5, respectively, within block k on machine M_1, and to $J^0_{21k}, J^1_{21k}, J_{2k}, J^2_{12k}, J^0_{12k}$ as groups 1, 2, 3, 4, 5, respectively, within block k on machine M_2. Also, we let v_{ik} denote the first job of group 4 of block k on machine M_i, for $i = 1, 2, k = 1, \ldots, \bar{s}$, where $v_{ik} = 0$ if the group is empty. The second property used by JBLOCK is:

(a) the jobs of groups 1 and 5 are sequenced according to the algorithm of Jackson (1956);

(b) the jobs of group 2 in $\{v_{11}, \ldots, v_{2\bar{s}}\}$ are sequenced consistently with $(v_{11}, \ldots, v_{2\bar{s}})$ before other jobs within that group, and these other jobs are then sequenced arbitrarily;

(c) the jobs of group 3 are sequenced arbitrarily;

(d) the jobs of group 4, with the exception of the first job, are sequenced arbitrarily.

The state variables in JBLOCK indicate for each machine, each group, and each block the total processing time of the operations that are assigned and the time at which processing starts. However, for group 3 we do not need the start time variables that are used to prevent the simultaneous processing of two operations of the same job. Additionally, a state variable is used to define the first job within group 4 on each machine and each block.

Algorithm JBLOCK is similar in structure to Algorithm BLOCK of §3.1. For the jobs in J_{12} and J_{21}, two operations are assigned to blocks, and a potential assignment is ignored if it corresponds to an overlap of two operations of the same job. The values of the state variables are restricted to ensure that there is no overlap of processing of two groups on the same machine. Having given the general structure of algorithm JBLOCK, we omit a formal description.

THEOREM 22. *Algorithm JBLOCK finds an optimal schedule for problem* $J2|s = \bar{s}|\hat{\gamma}$, *where* $\hat{\gamma}$ *is any objective in* $\mathcal{A}, \mathcal{B}, or \mathcal{D}$ *with time complexity* $O(n^{2\bar{s}-1}(\max\{\sum p_{1j}, \sum p_{2j}\})^{18\bar{s}-12})$, *and where* $\hat{\gamma}$ *is any objective in* $\mathcal{C}$ *with time complexity* $O(n^{2\bar{s}-1}(\max\{\sum p_{1j}, \sum p_{2j}\})^{18\bar{s}-10})$.

We note that JBLOCK can be generalized to provide a pseudopolynomial time algorithm for problems where the maximum number of operations of any job is a constant.

7. OPEN SHOPS

Recall that the recognition version of the classical problem $O2\|\sum C_j$ is unary *NP*-complete (Achugbue and Chin 1982). However, we now demonstrate that the corresponding fixed delivery date problem is only binary *NP*-complete. The binary *NP*-completeness result is shown using a reduction from Equal Cardinality Partition (Garey and Johnson 1979).

THEOREM 23. *The recognition version of problem* $O2|s = \bar{s}|\sum \widehat{C}_j$ *is binary* NP*-complete.*

We now describe the main features of a dynamic programming algorithm, OBLOCK, for several open shop problems with a constant number of fixed delivery dates. Each block k of a given schedule for problem $O2|s = \bar{s}|\hat{\gamma}$, where $\hat{\gamma}$ is any objective in $\mathcal{A}, \mathcal{B}, \mathcal{C}$, or $\mathcal{D}$, is partitioned into *groups* of jobs as follows. Let O^1_{12k} (respectively, O^1_{21k}) denote the set of jobs that have their first operation processed by machine M_1 (respectively, M_2) within block k and their second operation within a later block. Also, we let O^2_{12k} (respectively, O^2_{21k}) denote the set of jobs that have their second operation processed by machine M_2 (resp., M_1) within block k and their first operation within an earlier block. Finally, let O^0_{12k} (respectively, O^0_{21k}) denote the set of jobs that are processed first by machine M_1 (resp., M_2) and have both operations within block k.

Algorithm OBLOCK relies on various properties of an optimal schedule. First, the sequence of groups of block k on machine M_1 is $(O^0_{12k}, O^1_{12k}, O^2_{21k}, O^0_{21k})$, and the sequence of groups of block k on machine M_2 is $(O^0_{21k}, O^1_{21k}, O^2_{12k}, O^0_{12k})$, for $k = 1, \ldots, \bar{s}$. Accordingly, we refer to $O^0_{12k}, O^1_{12k}, O^2_{21k}, O^0_{21k}$ as groups 1, 2, 3, 4, respectively, within block k on machine M_1; and to $O^0_{21k}, O^1_{21k}, O^2_{12k}, O^0_{12k}$ as groups 1, 2, 3, 4, respectively, within block k on machine M_2. Also, we let v_{ik} denote the first job of group 3 of block k on machine M_i, for $i = 1, 2, k = 1, \ldots, \bar{s}$, where $v_{ik} = 0$ if the group is empty. The second property used by OBLOCK is:

(a) the jobs of groups 1 and 4 are sequenced according to the algorithm of Jackson (1956);

(b) the jobs of group 2 in $\{v_{11}, \ldots, v_{2\bar{s}}\}$ are sequenced consistently with $(v_{11}, \ldots, v_{2\bar{s}})$ before other jobs within that group, and these other jobs are then sequenced arbitrarily;

(c) the jobs of group 3, with the exception of the first job, are sequenced arbitrarily.

For the job shop studied in §6, all jobs with two operations are partitioned into sets J_{12} and J_{21} as part of the problem instance, whereas a partition of jobs into sets O_{12} and O_{21} for the open shop is not known until a solution is specified. To account for this extra generality in the open shop, we first index jobs according to the algorithm of Johnson (1954), which defines the processing order for groups O^0_{12k} on machine M_1 and O^0_{12k} on machine M_2 for jobs assigned to O_{12}. For jobs assigned to O_{21}, the order is reversed, which we implement by using a backward scheduling procedure. The state variables in OBLOCK are similar to those in JBLOCK. The algorithm itself is also similar. A minor reduction in complexity is achieved because there are only four groups here instead of five within each block on each machine.

THEOREM 24. *Algorithm OBLOCK finds an optimal schedule for problem* $O2|s = \bar{s}|\hat{\gamma}$, *where* $\hat{\gamma}$ *is any objective in* $\mathcal{A}, \mathcal{B}$, *or* $\mathcal{D}$ *with time complexity*

$O(n^{2\bar{s}-1}(\max\{\sum p_{1j}, \sum p_{2j}\})^{15\bar{s}-11})$, *and where* $\hat{\gamma}$ *is any objective in* $\mathscr{C}$ *with time complexity* $O(n^{2\bar{s}-1}(\max\{\sum p_{1j}, \sum p_{2j}\})^{15\bar{s}-9})$.

For the objective functions with due dates that we consider, Theorems 4 and 8 show that the corresponding problems with an arbitrary number of fixed delivery dates are less tractable. However, we now provide a further *NP*-completeness result, which uses a reduction from 3-Partition (Garey and Johnson 1979).

THEOREM 25. *The recognition version of problem* $O2|s|\sum \widehat{C}_j$ *is unary NP-complete.*

Some comments about problems $O3|s = \bar{s}|\widehat{C}_{\max}$ and $O3|s|\widehat{C}_{\max}$ appear in §8.

8. CONCLUDING REMARKS

This paper examines the solvability of scheduling problems in which jobs are dispatched to customers only at times that are fixed before the exact details of the schedule are determined. These problems are important in a variety of industrial applications that arise because of constraints on the resources needed for delivery. Moreover, there are several examples of a problem that has very different solvability when delivery dates are fixed, compared to the equivalent classical problem. These results, along with several practical considerations mentioned in the introduction, may lead to recommendations either for or against the use of a fixed delivery date schedule.

As shown in Table 1, we provide an almost complete "map" of the fixed delivery date problems that are defined by all the standard scheduling objectives and machine environments. Among the few open questions are the pseudopolynomial time solvability of problems $O3|s = \bar{s}|\widehat{C}_{\max}$ and $O3|s|\widehat{C}_{\max}$. However, the discovery of a pseudopolynomial time algorithm for either of those problems, and (unless $P = NP$) the contrapositive of Theorem 6, would imply a similar result for the classical problem $O3\|C_{\max}$. Also, a proof that the recognition version of either of those fixed delivery date problems is unary *NP*-complete, and (unless $P = NP$) the contrapositive of Theorem 5 would imply a similar result for problem $O3\|C_{\max}$. Thus, problems $O3|s = \bar{s}|\widehat{C}_{\max}$ and $O3|s|\widehat{C}_{\max}$ are exactly equivalent in complexity to $O3\|C_{\max}$, which is a well-known open problem (Lawler et al. 1993, Pinedo 1995). The only other open question is the pseudopolynomial time solvability of problem $Pm|s|\sum \widehat{T}_j$.

In view of the limited amount of earlier work on problems of this type, we believe this paper opens an important topic to future researchers in scheduling and in supply chain management. In particular, we believe that our demonstration in §2 of several techniques for proving general results in this area should be useful to other researchers. Some of the most important research issues for future study include the design of enumerative algorithms and of simple but effective heuristics, as well as improving the running time of our pseudopolynomial time algorithms. We hope that our work will stimulate research in these directions.

ACKNOWLEDGMENTS

This work was supported in part by NATO Collaborative Research Grant CRG 950773 and by the Summer Fellowship Program, Fisher College of Business, The Ohio State University. An associate editor and two anonymous referees provided helpful comments on an earlier draft of this paper.

REFERENCES

Achugbue, J. O., F. Y. Chin. 1982. Scheduling the open shop to minimize mean flow time. *SIAM J. Comput.* **11** 709–720.

Blum, N., R. W. Floyd, V. Pratt, R. L. Rivest, R. E. Tarjan. 1973. Time bounds for selection. *J. Comput. Systems Sci.* **7** 448–461.

Garey, M. R., D. S. Johnson. 1978. Strong *NP*-completeness results: motivation, examples and implications. *J. Assoc. Comput. Machinery* **25** 499–508.

——, ——. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

——, ——, R. Sethi. 1976. The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.* **1** 117–129.

Gonzalez, T., S. Sahni. 1976. Open shop scheduling to minimize finish time. *J. Assoc. Comput. Machinery* **23** 665–679.

Graham, R. L., E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan. 1979. Optimization and approximation in deterministic machine scheduling: a survey. *Ann. Discrete Math.* **5** 287–326.

Hall, N. G., M. Lesaoana, C. N. Potts. 1999. Scheduling with fixed delivery dates. The full text of this paper can be found at the Operations Research Home Page in the Online Collection. The Home Page can be accessed via ⟨www.informs.org⟩.

Jackson, J. R. 1995. Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science Research Project, University of California, Los Angeles.

——. 1956. An extension of Johnson's results on job lot scheduling. *Naval Res. Logist.* **3** 201–203.

Johnson, S. M. 1954. Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logist.* **1** 61–67.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations.* R. E. Miller, J. W. Thatcher (eds.), Plenum Press, New York, 85–103.

Lawler, E. L. 1977. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Ann. Discrete Math.* **1** 331–342.

——, J. K. Lenstra, A. H. G. Rinnooy Kan. 1981. Minimizing maximum lateness in a two-machine open-shop. *Math. Oper. Res.* **6**, 153–158; Erratum, 1982. *Math. Oper. Res.* **7** 635.

——, ——, ——, D. B. Shmoys. 1993. Sequencing and scheduling: algorithms and complexity. In *Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory*. S. C. Graves, A. H. G.

Rinnooy Kan, P. Zipkin eds. North-Holland, New York, 445–522.

——, J. M. Moore. 1969. A functional equation and its application to resource allocation and sequencing problems. *Management Sci.* **16** 77–84.

Lenstra, J. K. 1977. *Sequencing by Enumerative Methods*. Mathematical Centre Tract 69, Centre for Mathematics and Computer Science, Amsterdam.

Lesaoana, M. 1991. Scheduling with Fixed Delivery Dates. Ph.D. Thesis, University of Southampton, U.K.

Matsuo, H. 1988. The weighted total tardiness problem with fixed shipping times and overtime utilization. *Oper. Res.* **36** 293–307.

Moore, J. M. 1968. An *n* job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Sci.* **15** 102–109.

Pinedo, M. 1995. *Scheduling: Theory, Algorithms and Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Schonhäge, A., M. Patterson, N. Pippenger. 1976. Finding the median. *J. Comput. Systems Sci.* **13** 184–199.

Smith, W. E. 1956. Various optimizers for single stage production. *Naval Res. Logist. Quart.* **3** 59–66.

Williamson, D. P., L. A. Hall, J. A. Hoogeveen, C. A. J. Hurkens, J. K. Lenstra, S. V. Sevast'janov, D. B. Shmoys. 1997. Short shop schedules. *Oper. Res.* **45** 288–294.